

**UNITED STATES PATENT APPLICATION FOR**


**LOAD BALANCING IN A STORAGE NETWORK**

Inventors:  
Santosh C. Lolayekar  
Yu-Ping Cheng

**CERTIFICATE OF MAILING BY "EXPRESS MAIL"  
UNDER 37 C.F.R. § 1.10**

"Express Mail" mailing label number: EL67072888US  
Date of Mailing: January 18, 2002

I hereby certify that this correspondence is being deposited with the United States Postal Service, utilizing the "Express Mail Post Office to Addressee" service addressed to **Box PATENT APPLICATION, Assistant Commissioner for Patents, Washington, D.C. 20231** and mailed on the above Date of Mailing with the above "Express Mail" mailing label number.



---

Johann S. Mercado  
Signature Date: January 18, 2002

LOAD BALANCING IN A STORAGE NETWORK

Inventors:

Santosh C. Lolayekar  
Yu-Ping Cheng

5

10

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to Provisional Application Serial No. 60/325,704, entitled STORAGE SWITCH FOR STORAGE AREA NETWORK, and filed September 28, 2001, and incorporated by reference herein.

15

[0002] This application is also related to the following applications, all filed concurrently herewith and all incorporated herein by reference:

STORAGE SWITCH FOR STORAGE AREA NETWORK,

20

Serial No. \_\_\_\_\_ [atty. dkt. No. MARA-01000US1];

PROTOCOL TRANSLATION IN A STORAGE SYSTEM,

Serial No. \_\_\_\_\_ [atty. dkt. No. MARA-01001US0];

SERVERLESS STORAGE SERVICES,

Serial No. \_\_\_\_\_ [atty. dkt. No. MARA-01002US0];

25

PACKET CLASSIFICATION IN A STORAGE SYSTEM,

Serial No. \_\_\_\_\_ [atty. dkt. No. MARA-01003US0];

VIRTUALIZATION IN A STORAGE SYSTEM,

Serial No. \_\_\_\_\_ [atty. dkt. No. MARA-01005US0];

ENFORCING QUALITY OF SERVICE IN A STORAGE NETWORK

30

Serial No. \_\_\_\_\_ [atty. dkt. No. MARA-01006US0]; and

POOLING AND PROVISIONING STORAGE RESOURCES

IN A STORAGE NETWORK,

Serial No. \_\_\_\_\_ [atty. dkt. No. MARA-01007US0].

FIELD OF INVENTION

[0003] The invention generally relates to storage area networks.

BACKGROUND

5 [0004] The rapid growth in data intensive applications continues to fuel the demand for raw data storage capacity. As companies rely more and more on e-commerce, online transaction processing, and databases, the amount of information that needs to be managed and stored can be massive. As a result, the ongoing need to add more storage, service more users, and back-up more data has become a  
10 daunting task.

[0005] To meet this growing demand for data, the concept of the Storage Area Network (SAN) has been gaining popularity. A SAN is defined by the Storage Networking Industry Association (SNIA) as a network whose primary purpose is the transfer of data between computer systems and storage elements and  
15 among storage elements. Unlike connecting a storage device directly to a server, e.g., with a SCSI connection, and unlike adding a storage device to a LAN with a traditional interface such as Ethernet (e.g., a NAS system), the SAN forms essentially an independent network that does not tend to have the same bandwidth limitations as its direct-connect SCSI and NAS counterparts and also provides  
20 increased configurability and scalability.

[0006] More specifically, in a SAN environment, storage devices (e.g., tape drives and RAID arrays) and servers are generally interconnected via various switches and appliances. The connections to the switches and appliances are usually Fibre Channel. This structure generally allows for any server on the SAN  
25 to communicate with any storage device and vice versa. It also provides alternative paths from server to storage device. In other words, if a particular server is slow or completely unavailable, another server on the SAN can provide access to the storage device. A SAN also makes it possible to mirror data, making multiple copies available and thus creating more reliability in the availability of data. When  
30 more storage is needed, additional storage devices can be added to the SAN

without the need to be connected to a specific server; rather, the new devices can simply be added to the storage network and can be accessed from any point.

[0007] An example of a SAN is shown in the system 100 illustrated in the functional block diagram of Fig. 1. As shown, there are one or more servers 102.

5 Three servers 102 are shown for exemplary purposes only. Servers 102 are connected through an Ethernet connection to a LAN 106 and/or to a router 108 and then to a WAN 110, such as the Internet. In addition, each server 102 is connected through a Fibre Channel connection to each of a plurality of Fibre Channel switches 112 sometimes referred to as the "fabric" of the SAN. Two switches 112  
10 are shown for exemplary purposes only. Each switch 112 is in turn connected to each of a plurality of SAN appliances 114. Two appliances 114 are shown for exemplary purposes only. Each appliance is also coupled to each of a plurality of storage devices 116, such as tape drives, optical drives, or RAID arrays. In addition, each switch 112 and appliance 114 is coupled to a gateway 118, which  
15 in turn is coupled to router 108, which ultimately connects to a Wide Area Network (WAN) 118, such as the Internet. Fig. 1 shows one example of a possible configuration of a SAN 119, which includes switches 112, appliances 114, storage devices 116, and gateways 118. Still other configurations are possible. For instance, one appliance may be connected to fewer than all the switches.

20 [0008] Appliances 114 perform the storage management of the SAN. When the appliance 114 receives data, it stores the data in a memory in the appliance. Then, with a processor (also in the appliance), analyzes and operates on the data in order to forward the data to the correct storage device(s). This store-and-forward process typically slows down data access.

25 [0009] While the appliances do perform some switching, because there may be a large number of servers (many more than three), and because each appliance has few ports (usually only two or four), switches 112 are needed to connect the many servers to the few appliances. Nevertheless, switches 112 have little built-in intelligence and merely forward data to a selected appliance 114. One limitation  
30 of appliances is the fact that many appliances often have a limited or set number

of ports. Adding ports to an appliance, although possible, is typically very expensive. Every one or two ports are supported by an expensive CPU or server card. So generally to add ports, entire file cards (which perform virtualization and store-and-forward functions) must be added to the device, which is usually very costly. In the alternative, appliances are simply added to the SAN, but again, this tends to be very costly.

[0010] In addition, SANs, usually in the appliances 114, generally perform a function known as "virtualization." Virtualization occurs when space on one or more physical storage devices is allocated to a particular user, but the physical location of that space remains unknown to the user. For instance, a user may access its company's "engineering storage space," ENG:, accessing and "seeing" the virtual space ENG: as he or she would access or "see" an attached disk drive. Nonetheless, the ENG: space may be divided over several physical storage devices or even fragmented on a single storage device. Thus, when a server requests a virtual device (e.g., ENG:) and block number, the appliance must determine the device(s) that physically correlate to the virtual device requested and direct the data accordingly.

[0011] Although SANs were introduced several years ago, interoperability problems, lack of available skills, and high implementation costs remain major obstacles to widespread use. For instance, SANs as they currently exist have high deployment costs and high management costs. Referring again to Fig. 1, each switch, appliance, and gateway typically come from different vendors, creating a lack of management standards that has resulted in the proliferation of vendor-specific management tools. As a result, to deploy a SAN, equipment must be purchased from multiple vendors. And, as shown in Fig. 1, each switch, appliance, gateway, storage device, server, and router will have its own management, shown as management stations 120. Although independent physical management stations are shown, it is to be understood that independent management is frequently in the form of independent, vendor-specific software on a single computer but which software does not communicate with one another. As a result, there is no

centralized management of the SAN and its management costs are high given that there are usually multiple management stations that frequently require many people to manage.

[0012] In addition, "provisioning" of (or "creating") virtual targets for  
5 SANs has become burdensome. When a new virtual target needs to be created, a human administrator must first determine the application requirements for the data, such as performance, capacity required initially plus that required for potential growth, data availability, and data protection. More specifically, the administrator must allocate all or part of one or more physical devices to the virtual target and  
10 configure those devices to produce the best performance as well as access control for data security. The administrator must further assure the routes through the storage network have the level of availability required and may have to install alternate pathing if high availability is required so that if one path goes down another path to the target is available. Finally, the administrator must test the  
15 environment to verify the functionality before making the virtual target accessible. Overall, it may take several days or even weeks to create such a virtual target – a time period that is often unacceptable to users of the SAN.

#### SUMMARY

20 [0013] A system in accordance with an embodiment of the invention automatically discovers storage resources in communication with a switch and obtains information about the characteristics of those resources. Once the characteristics are known, in one embodiment, the device is classified according to a predefined policy and then placed in a storage pool.

25 [0014] From the pool a virtual target can be provisioned. In one embodiment the virtual target is placed in a user domain. An initiator connection is also provisioned in one embodiment. The virtual target, the initiator connection, and the user domain all serve in one embodiment to define a Quality of Service (QoS) policy.

[0015] A system in accordance with another embodiment of the invention can further enforce Quality of Service for connections between initiators and targets. Quality of Service, in one embodiment, is enforced by controlling the number of concurrent requests that can be sent from an initiator to a target.

[0016] A system in accordance with still another embodiment of the invention can dynamically provide load balancing. In one embodiment, load balancing is performed by sending requests on one of a plurality of alternate paths to a target where the path selected has the shortest average response time. In another embodiment, load balancing occurs in mirrored targets where a request is sent to the member of the mirrored target with the shortest average response time.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0017] The present invention is described with respect to particular exemplary embodiments thereof and reference is accordingly made to the drawings in which:

Fig. 1 is a generalized function block diagram of a SAN in accordance with a conventional system;

Fig. 2 is a generalized function block diagram of a SAN system using a storage switch in accordance with an embodiment of the invention;

Fig. 3 is a generalized function block diagram of another embodiment of a system using a storage switch in accordance with an embodiment of the invention;

Fig. 4 is a generalized function block diagram of yet another embodiment of a system using a storage switch in accordance with an embodiment of the invention;

Fig. 5 is a generalized function block diagram of a storage switch in accordance with an embodiment of the invention;

Fig. 6 is a generalized function block diagram of a linecard used in a storage switch in accordance with an embodiment of the invention;

Fig. 7a is a generalized block diagram of a Virtual Target Descriptor used in a storage switch in accordance with an embodiment of the invention;

5 Fig. 7b is a generalized block diagram of a Physical Target Descriptor used in a storage switch in accordance with an embodiment of the invention;

Fig. 8 is a generalized block diagram illustrating storage pools;

Fig. 9 is a generalized logic block diagram illustrating virtual targets as “seen” by a server;

10 Fig. 10a is a generalized block diagram illustrating exemplary storage pools of physical devices;

Figs. 10b-10d are generalized block diagrams illustrating various exemplary virtual target storage pools;

15 Fig. 11 is a generalized block diagram illustrating the accessibility from a first switch of a storage device coupled to a second switch;

Fig. 12 is a flow diagram illustrating steps in accordance with an embodiment of the invention; and

Figs. 13a-13b illustrate, with generalized block diagrams, load balancing.

20

#### DETAILED DESCRIPTION

[0018] A system 200 that includes a storage switch in accordance with the invention is illustrated in Fig. 2. As shown, such a system is greatly simplified over existing systems. In one embodiment, system 200 includes a plurality of servers 202. For purposes of illustration only, three servers 202 are shown, although more or fewer servers could be used in other embodiments. Although not shown, the servers could also be coupled to a LAN. As shown, each server 202 is connected to a storage switch 204. In other embodiments, however, each server 202 may be connected to fewer than all of the storage switches 204 present. The connections formed between the servers and

25

30

switches can utilize any protocol, although in one embodiment the connections are either Fibre Channel or Gigabit Ethernet (carrying packets in accordance with the iSCSI protocol). Other embodiments may use the Infiniband protocol, defined by Intel Inc., or other protocols or connections.

5     **[0019]**         In the embodiment illustrated, each switch 204 is in turn connected to each of a plurality of storage devices or subsystems 206. Nonetheless, in other embodiments, each switch 204 may be connected to fewer than all of the storage devices or subsystems 206. The connections formed between the storage switches 204 and storage devices 206 can utilize any  
10     protocol, although in one embodiment the connections are either Fibre Channel or Gigabit Ethernet.

**[0020]**         In some embodiments, one or more switches 204 are each coupled to a Metropolitan Area Network (MAN) or Wide Area Network (WAN) 208, such as the Internet. The connection formed between a storage  
15     switch 204 and a WAN 208 will generally use the Internet Protocol (IP) in most embodiments. Although shown as directly connected to MAN/WAN 208, other embodiments may utilize a router (not shown) as an intermediary between switch 204 and MAN/WAN 208.

**[0021]**         In addition, respective management stations 210 are connected to  
20     each storage switch 204, to each server 202, and to each storage device 206. Although management stations are illustrated as distinct computers, it is to be understood that the software to manage each type of device could collectively be on a single computer.

**[0022]**         Fig. 3 shows an alternative embodiment of a system in  
25     accordance with the invention. In such an embodiment, two SANs 302, 304 are formed, each using one or more storage switches 204 in accordance with an embodiment of the invention. The SANs 302 and 304 are coupled through a WAN 208, such as the Internet, by way of switches 204. Connections 208 can be any standard or protocol, but in one embodiment will be Packet over SONET  
30     (PoS) or 10 Gigabit Ethernet.

**[0023]** Fig. 4 shows still another embodiment of a system in accordance with the invention wherein switches 204 are coupled directly to one another. In any of the embodiments shown in Figs. 2 or 3, if more than one switch is used, those switches could be coupled as illustrated in Fig. 4.

5 **[0024]** A storage switch in accordance with the invention enables a centralized management of globally distributed storage devices, which can be used as shared storage pools, instead of having a huge number of management stations distributed globally and an army of skilled management personnel. Such a storage switch is an “intelligent” switch, and, as can be seen by comparing  
10 Fig. 2 to Fig. 1, the functions of switch, appliance, and gateway have effectively been united in a storage switch 204 in accordance with an embodiment of the invention. Such a storage switch 204, in addition to its switching function, provides the virtualization and storage services (e.g., mirroring) that would typically be provided by appliances in conventional architectures, and it also  
15 provides protocol translation. A storage switch in accordance with some embodiments of the invention also performs additional functions (for instance, data security through a Virtual Private Network). Such additional functions include functions that are performed by other devices in conventional systems, such as load balancing, which is traditionally performed by the servers, as well  
20 as other functions not previously available in conventional systems, such as Quality of Service for storage access. Moreover, in one embodiment the Quality of Service for storage access function is “application aware” — that is, the Quality of Service provided is specified by the nature of the application initiating a connection to a storage target.

25 **[0025]** In addition, the intelligence of a storage switch in accordance with an embodiment of the invention is distributed to every switch port. This distributed intelligence allows for system scalability and availability.

**[0026]** Further, the distributed intelligence allows a switch in accordance with an embodiment of the invention to process data at “wire  
30 speed,” meaning that a storage switch 204 introduces no more latency to a data

packet than would be introduced by a typical network switch (such as switch 112 in Fig. 1). Thus, “wire speed” for the switch is measured by the connection to the particular port. Accordingly, in one embodiment having OC-48 connections, the storage switch can keep up with an OC-48 speed (2.5 bits per ns). A two Kilobyte packet (with 10 bits per byte) moving at OC-48 speed takes as little as eight microseconds coming into the switch. A one Kilobyte packet takes as little as four microseconds. A minimum packet of 100 bytes only elapses merely 400 ns. Nonetheless, when the term “wire-speed” processing is used herein, it does not mean that such processing needs as few as 400 ns to process a 100-byte packet. However, it does mean that the storage switch can handle the maximum Ethernet packet of 1500 bytes (with ten-bit encoding, so that a byte is ten bits) at OC-48 speed, i.e., in about 6  $\mu$ s (4  $\mu$ s per Kilobyte or 2.5 bits per ns), in one embodiment. In embodiments with a 1 Gb Ethernet port, where processing is generally defined as one bit per nanosecond, “wire-speed” data for that port will be 10  $\mu$ s per Kilobyte, indicating that the switch has up to 10  $\mu$ s to process a Kilobyte. In embodiments with a 2 Gb Fibre Channel port, “wire speed” will be 5  $\mu$ s per Kilobyte. Still other embodiments may process data at ten Gigabit Ethernet or OC-192 speeds or faster.

[0027] As used herein, “virtualization” essentially means the mapping of a virtual target space subscribed to by a user to a space on one or more physical storage target devices. The terms “virtual” and “virtual target” come from the fact that storage space allocated per subscription can be anywhere on one or more physical storage target devices connecting to a storage switch 204. The physical space can be provisioned as a “virtual target” which may include one or more “logical units” (LUs). Each virtual target consists of one or more LUs identified with one or more LU numbers (LUNs), which are frequently used in the iSCSI and FC protocols. Each logical unit is generally comprised of one or more extents – a contiguous slice of storage space on a physical device. Thus, a virtual target may occupy a whole storage device (one extent), a part of a single storage device (one or more extents), or parts of multiple storage devices

(multiple extents). The physical devices, the LUs, the number of extents, and their exact locations are immaterial and invisible to a subscriber user.

**[0028]** While the storage space may come from a number of different physical devices, each virtual target belongs to one or more “pools,” sometimes referred to herein as “domains.” Only users of the same domain are allowed to share the virtual targets in their domain. Domain-sets can also be formed that include several domains as members. Use of domain-sets can ease the management of users of multiple domains, e.g., if one company has five domains but elects to discontinue service, only one action need be taken to disable the domain-set as a whole. The members of a domain-set can be members of other domains as well.

**[0029]** Fig. 5 illustrates a function block diagram of a storage switch 204 in accordance with an embodiment of the invention. In one embodiment, the storage switch 204 includes a plurality of linecards 502, 504, and 506, a plurality of fabric cards 508, and two system control cards 510, each of which will be described in further detail below.

**[0030]** System Control Cards. Each of the two System Control Cards (SCCs) 510 connects to every line card 502, 504, 506. In one embodiment, such connections are formed by I<sup>2</sup>C signals, which are well known in the art, and through an Ethernet connection with the SCC. The SCC controls power up and monitors individual linecards, as well as the fabric cards, with the I<sup>2</sup>C connections. Using inter-card communication over the ethernet connections, the SCC also initiates various storage services, e.g., snapshot and replicate, discussed in Provisional Application No. 60/325,704.

**[0031]** In addition the SCC maintains a database 512 that tracks configuration information for the storage switch as well as all virtual targets and physical devices attached to the switch, e.g., servers and storage devices. In addition, the database keeps information regarding usage, error and access data, as well as information regarding different domains and domain sets of virtual targets and users. The records of the database are referred to herein as

“objects.” Each initiator (e.g., a server) and target (e.g., a storage device) has a World Wide Unique Identifier (WWUI), which are known in the art. The database is maintained in a memory device within the SCC, which in one embodiment is formed from flash memory, although other memory devices will also be satisfactory.

[0032] The storage switch 204 can be reached by a management station 210 through the SCC 510 using an ethernet connection. Accordingly, the SCC also includes an additional Ethernet port for connection to a management station. An administrator at the management station can discover the addition or removal of storage devices or virtual targets, as well as query and update virtually any object stored in the SCC database 512.

[0033] Of the two SCCs 510, one is the main operating SCC while the other is a backup, remaining synchronized to the actions in the storage switch, but not directly controlling them. The SCCs operate in a high availability mode wherein if one SCC fails, the other becomes the primary controller.

[0034] Fabric Cards. In one embodiment of switch 204, there are three fabric cards 508, although other embodiments could have more or fewer fabric cards. Each fabric card 508 is coupled to each of the linecards 502, 504, 506 in one embodiment and serves to connect all of the linecards together. In one embodiment, the fabric cards 508 can each handle maximum traffic when all linecards are populated. Such traffic loads handled by each linecard are up to 160 Gbps in one embodiment although other embodiments could handle higher or lower maximum traffic volumes. If one fabric card 508 fails, the two surviving cards still have enough bandwidth for the maximum possible switch traffic: in one embodiment, each linecard generates 20 Gbps of traffic, 10 Gbps ingress and 10 Gbps egress. However, under normal circumstances, all three fabric cards are active at the same time. From each linecard, the data traffic is sent to any one of the three fabric cards that can accommodate the data.

[0035] Linecards. The linecards form connections to servers and to storage devices. In one embodiment, storage switch 204 supports up to sixteen

linecards although other embodiments could support a different number.

Further, in one embodiment, three different types of linecards are utilized:

Gigabit Ethernet (GigE) cards 502, Fibre Channel (FC) cards 504, and WAN cards 506. Other embodiments may include more or fewer types of linecards.

5 The GigE cards 502 are for Ethernet connections, connecting in one embodiment to either iSCSI servers or iSCSI storage devices (or other Ethernet based devices). The FC cards 504 are for Fibre Channel connections, connecting to either Fibre Channel Protocol (FCP) servers or FCP storage devices. The WAN cards 506 are for connecting to a MAN or WAN.

10 [0036] Fig. 6 illustrates a functional block diagram of a generic line card 600 used in one embodiment of a storage switch 204 in accordance with the invention. The illustration shows those components that are common among all types of linecards, e.g., GigE 502, FC 504, or WAN 506. In other embodiments other types of linecards can be utilized to connect to devices using other  
15 protocols, such as Infiniband. The differences in the linecards are discussed subsequently.

[0037] Ports. Each line card 600 includes a plurality of ports 602. The ports form the linecard's connections to either servers or storage devices. Eight ports are shown in the embodiment illustrated, but more or fewer  
20 could be used in other embodiments. For example, in one embodiment each GigE card can support up to eight 1Gb Ethernet ports, each FC card can support up to either eight 1Gb FC ports or four 2Gb FC ports, and each WAN card can support up to four OC-48 ports or two OC-192 ports. Thus, in one embodiment, the maximum possible connections are 128 ports per switch 204. The ports of  
25 each linecard are full duplex and connect to either a server or other client, or to a storage device or subsystem.

[0038] In addition each port 602 has an associated memory 603. Although only one memory device is shown connected to one port, it is to be understood that each port may have its own memory device or the ports may all

be coupled to a single memory device. Only one memory device is shown here coupled to one port for clarity of illustration.

[0039] Storage Processor Unit. In one embodiment, each port is associated with a Storage Processor Unit (SPU) 601. In one embodiment the SPU rapidly processes the data traffic allowing for wire-speed operations. In one embodiment, the SPU includes several elements: a Packet Aggregation and Classification Engine (PACE) 604, a Packet Processing Unit (PPU) 606, an SRAM 605, and a CAM 607. Still other embodiments may use more or fewer elements or could combine elements to obtain the same functionality. For instance, some embodiments may include a PACE and a PPU in the SPU, but the SPU may share memory elements with other SPUs.

[0040] PACE. Each port is coupled to a Packet Aggregation and Classification Engine (PACE) 604. As illustrated, the PACE 604 aggregates two ports into a single data channel having twice the bandwidth. For instance, the PACE 604 aggregates two 1Gb ports into a single 2Gb data channel. The PACE classifies each received packet into a control packet or a data packet, as described in Provisional Application No. 60/325,704. Control packets are sent to the CPU 614 for processing, via bridge 616. Data packets are sent to a Packet Processing Unit (PPU) 606, discussed below, with a local header added. In one embodiment the local header is sixteen bytes resulting in a data "cell" of 64 bytes (16 bytes of header and 48 bytes of payload). The local header is used to carry information and used internally by switch 204. The local header is removed before the packet leaves the switch. Accordingly, as used herein a "cell" is a transport unit that is used locally in the switch that includes a local header and the original packet (in some embodiments, the original TCP/IP headers are also stripped from the original packet). Nonetheless, not all embodiments of the invention will create a local header or have "internal packets" (cells) that differ from external packets. Accordingly, the term "packet" as used herein can refer to either "internal" or "external" packets.

[0041] The classification function helps to enable a switch to perform storage virtualization and protocol translation functions at wire speed without using a store-and-forward model of conventional systems. Each PACE has a dedicated path to a PPU 606 while all four PACEs in the illustrated embodiment share a path to the CPU 614, which in one embodiment is a 104MHz/32 (3.2 Gbps) bit data path.

[0042] Packet Processing Unit (PPU). The PPU 606 performs virtualization and protocol translation on-the-fly, meaning, the cells are not buffered for such processing, as described in Provisional Application No. 60,325,704. It also implements other switch-based storage service functions, described later. The PPU is capable, in one embodiment, of moving cells at OC-48 speed or 2.5 Gbps for both the ingress and egress directions, while in other embodiments it can move cells at OC-192 speeds or 10 Gbps. The PPU in one embodiment includes an ingress PPU 606<sub>i</sub> and an egress PPU 606<sub>e</sub>, which both run concurrently. The ingress PPU 606<sub>i</sub> receives incoming data from PACE 604 and sends data to the Traffic Manager 608<sub>i</sub> while the egress PPU 606<sub>e</sub> receives data from Traffic Manager 608<sub>e</sub> and sends data to a PACE 604. Although only one PPU 606 is shown in Fig. 6 as having an ingress PPU 606<sub>i</sub> and an egress PPU 606<sub>e</sub>, it is to be understood that in one embodiment all PPUs 606 will include both an ingress and an egress PPU and that only one PPU is shown in Fig. 6 with both ingress and egress PPUs for clarity of illustration.

[0043] A large number of storage connections (e.g., server to virtual target) can be established concurrently at each port. Nonetheless, each connection is unique to a virtual target and can be uniquely identified by a TCP Control Block Index (in the case of iSCSI connections) and a port number. When a connection is established, the CPU 614 of the linecard 600 informs the PPU 606 of an active virtual target by sending it a Virtual Target Descriptor (VTD) for the connection. The VTD includes all relevant information regarding the connection and virtual target that the PPU will need to properly operate on

the data, e.g., perform virtualization, translation, and various storage services. The VTD is derived from an object in the SCC database and usually contains a subset of information that is stored in the associated object in the SCC database. An example of the fields in a VTD in one embodiment of the invention are shown in Fig. 7a. Nonetheless, other embodiments of the invention may have a VTD with more, fewer, or different fields.

[0044] Similarly, Physical Target Descriptors (PTDs) are utilized in an embodiment of the invention. PTDs describe the actual physical devices, their individual LUs, or their individual extents (a contiguous part of or whole LU) and will include information similar to that for the VTD. Also, like the VTD, the PTD is derived from an object in the SCC database. An example of the fields in a PTD in one embodiment of the invention are shown in Fig. 7b. Nonetheless, other embodiments of the invention may have a PTD with more, fewer, or different fields.

[0045] To store the VTDs and PTDs and have quick access to them, in one embodiment the PPUs 606 are connected to an SRAM 605 and CAM 607. SRAM 605 stores a VTD and PTD database. A listing of VTD Identifiers (VTD IDs), or addresses, as well as PTD Identifiers (PTD IDs), is also maintained in the PPU CAM 607 for quick accessing of the VTDs. The VTD IDs are indexed (mapped) using a TCP Control Block Index and a LUN. The PTD IDs are indexed using a VTD ID. In addition, for IP routing services, the CAM 607 contains a route table, which is updated by the CPU when routes are added or removed.

[0046] Note that although only one CAM and an SRAM are illustrated as connected to one PPU, this is to maintain clarity of the illustration. In various embodiments, each PPU will be connected with its own CAM and SRAM device, or the PPUs will all be connected to a single CAM and/or SRAM.

[0047] For each outstanding request to the PPU (e.g., reads or writes), a task control block is established in the PPU SRAM 607 to track the status of the

request. There are ingress task control blocks (ITCBs) tracking the status of requests received by the storage switch on the ingress PPU and egress task control blocks (ETCBs) tracking the status of requests sent out by the storage switch on the egress PPU. For each virtual target connection, there can be a large number of concurrent requests, and thus many task control blocks. Task control blocks are allocated as a request begins and freed as the request completes.

[0048] Traffic Manager. There are two traffic managers (TMs) 608 on each linecard 600: one TM 608<sub>i</sub> for ingress traffic and one TM 608<sub>e</sub> for egress traffic. The ingress TM receives cells from all four SPUs, in the form of 64-byte data cells, in one embodiment. In such an embodiment, each data cell has 16 bytes of local header and 48 bytes of payload. The header contains a FlowID that tells the TM the destination port of the cell. In some embodiments, the SPU may also attach a TM header to the cell prior to forwarding the cell to the TM. Either the TM or the SPU can also subdivide the cell into smaller cells for transmission through the fabric cards in some embodiments.

[0049] The ingress TM sends data cells to the fabric cards via a 128-bit 104 Mhz interface 610 in one embodiment. Other embodiments may operate at 125 Mhz or other speeds. The egress TM receives the data cells from the fabric cards and delivers them to the four SPUs.

[0050] Both ingress and egress TMs have a large buffer 612 to queue cells for delivery. Both buffers 612 for the ingress and egress TMs are 64MB, which can queue a large number of packets. The SPUs can normally send cells to the ingress TM quickly as the outgoing flow of the fabric cards is as fast as the incoming flow. Hence, the cells are moving to the egress TM quickly. On the other hand, an egress TM may be backed up because the outgoing port is jammed or being fed by multiple ingress linecards. In such a case, a flag is set in the header of the outgoing cells to inform the egress SPU to take actions quickly. The egress TM also sends a request to the ingress SPU to activate a flow control function, discussed further below, used in providing Quality of

Service for Storage access. It is worth noting that, unlike communications traffic over the Internet, for storage traffic dropping a packet or cell is unacceptable. Therefore, as soon as the amount of cells in the buffer exceeds a specified threshold, the SPU must activate its flow control function to slow down the incoming traffic to avoid buffer overflow.

5 [0051] Fabric Connection. The fabric connection 610 converts the 256-bit parallel signals of the TM (128 bits ingress and 128 bits egress, respectively), into a 16-bit serial interface (8-bit ingress and 8-bit egress) to the backplane at 160 Gbps. Thus the backplane is running at one sixteenth of the  
10 pins but sixteen times faster in speed. This conversion enables the construction of a high availability backplane at a reasonable cost without thousands of connecting pins and wires. Further, because there are three fabric cards in one embodiment, there are three high-speed connectors on each linecard in one embodiment, wherein the connectors each respectively connect the 8-bit signals  
15 to a respective one of the three fabric cards. Of course, other embodiments may not require three fabric connections 610.

[0052] CPU. On every linecard there is a processor (CPU) 614, which in one embodiment is a PowerPC 750 Cxe. In one embodiment, CPU 614 connects to each PACE with a 3.2 Gb bus, via a bus controller 615 and a  
20 bridge 616. In addition, CPU 614 also connects to each PPU, CAM and TM, however, in some embodiments this connection is slower at 40 Mbps. Both the 3.2 Gb and 40 Mb paths allow the CPU to communicate with most devices in the linecard as well as to read and write the internal registers of every device on the linecard, download microcode, and send and receive control packets.

25 [0053] The CPU on each linecard is responsible to initialize every chip at power up and to download microcode to the SPUs and each port wherever the microcode is needed. Once the linecard is in running state, the CPU processes the control traffic. For information needed to establish a virtual target connection, the CPU requests the information from the SCC, which in turn gets  
30 the information from an appropriate object in the SCC database.

[0054]        Distinction in Linecards - Ports. The ports in each type of linecard, e.g., GigE, FC, or WAN are distinct as each linecard only supports one type of port in one embodiment. Each type of port for one embodiment is described below. Of course other linecard ports could be designed to support other protocols, such as Infiniband in other embodiments.

[0055]        GigE Port. A gigabit Ethernet port connects to iSCSI servers and storage devices. While the GigE port carries all kinds of Ethernet traffic, the only network traffic generally to be processed by a storage switch 204 at wire speed in accordance with one embodiment of the invention is an iSCSI Packet Data Unit (PDU) inside a TCP/IP packet. Nonetheless, in other embodiments packets in accordance with other protocols (like Network File System (NFS)) carried over Ethernet connections may be received at the GigE Port and processed by the SPU and/or CPU.

[0056]        The GigE port receives and transmits TCP/IP segments for virtual targets or iSCSI devices. To establish a TCP connection for a virtual target, both the linecard CPU 614 and the SCC 510 are involved. When a TCP packet is received, and after initial handshaking is performed, a TCP control block is created and stored in the GigE port memory 603. A VTD must also be retrieved from an object of the SCC database and stored in the CPU SDRAM 605 for the purpose of authenticating the connection and understanding the configuration of the virtual target. The TCP Control Block identifies a particular TCP session or iSCSI connection to which the packet belongs, and contains in one embodiment, TCP segment numbers, states, window size, and potentially other information about the connection. In addition, the TCP Control Block is identified by an index, referred to herein as the "TCP Control Block Index." A VTD for the connection must be created and stored in the SPU SRAM 605. The CPU creates the VTD by retrieving the VTD information stored in its SDRAM and originally obtained from the SCC database. A VTD ID is established in a list of VTD IDs in the SPU CAM 607 for quick reference

to the VTD. The VTD ID is affiliated with and indexed by the TCP Control Block Index.

[0057] When the port receives iSCSI PDUs, it serves essentially as a termination point for the connection, but then the switch initiates a new connection with the target. After receiving a packet on the ingress side, the port delivers the iSCSI PDU to the PACE with a TCP Control Block Index, identifying a specific TCP connection. For a non-TCP packet or a TCP packet not containing an iSCSI PDU, the port receives and transmits the packet without acting as a termination point for the connection. Typically, the port 602 communicates with the PACE 604 that an iSCSI packet is received or sent by using a TCP Control Block Index. When the TCP Control Block Index of a packet is -1, it identifies a non-iSCSI packet.

[0058] FC Port. An FC port connects to servers and FC storage devices. The FC port appears as a fibre channel storage subsystem (i.e., a target) to the connecting servers, meaning, it presents a large pool of virtual target devices that allow the initiators (e.g., servers) to perform a Process Login (PLOGI or PRLI), as are understood in the art, to establish a connection. The FC port accepts the GID extended link services (ELs) and returns a list of target devices available for access by that initiator (e.g., server).

[0059] When connecting to fibre channel storage devices, the port appears as a fibre channel F-port, meaning, it accepts a Fabric Login, as is known in the art, from the storage devices and provides name service functions by accepting and processing the GID requests — in other words, the port will appear as an initiator to storage devices.

[0060] In addition, an FC port can connect to another existing SAN network, appearing in such instances as target with many LUs to the other network.

[0061] At the port initialization, the linecard CPU must go through both sending Fabric Logins, Process Logins, and GIDs as well as receive the same.

The SCC supports an application to convert FC ELS's to iSNS requests and

responses. As a result, the same database in the SCC keeps track both the FC initiators (e.g., servers) and targets (e.g., storage devices) as if they were iSCSI initiators and targets.

5 [0062] When establishing an FC connection, unlike for a GigE port, an FC port does not need to create TCP control blocks or their equivalent; all the necessary information is available from the FC header. But, a VTD (indexed by a D\_ID) will still need to be established in a manner similar to that described for the GigE port.

10 [0063] An FC port can be configured for 1Gb or 2Gb. As a 1Gb port, two ports are connected to a single PACE as illustrated in Fig. 6; but in an embodiment where it is configured as a 2Gb port, port traffic and traffic that can be accommodated by the SPU should match to avoid congestion at the SPU. The port connects to the PACE with a POS/PHY interface in one embodiment. Each port can be configured separately, i.e. one PACE may have two 1 Gb ports and another PACE has a single 2 Gb port.

15 [0064] WAN Ports. In embodiments that include a WAN linecard, the WAN linecard supports OC-48 and OC-192 connections in one embodiment. Accordingly, there are two types of WAN ports: OC-48 and OC-192. For OC-48, there is one port for each SPU. There is no aggregation function in the PACE, although there still is the classification function. A WAN port connects to SONET and works like a GigE port as it transmits and receives network packets such as ICMP, RIP, BPG, IP and TCP. Unlike the GigE port, a WAN port in one embodiment supports network security with VPN and IPsec that requires additional hardware components.

25 [0065] Since OC-192 results in a faster wire speed, a faster SPU will be required in embodiments that support OC-192.

Switch-Based Storage Operations

5 [0066] A storage switch in accordance with an embodiment of the invention performs various switch-based storage operations, including pooling and provisioning, Quality of Service for storage access, and load balancing, each of which will be discussed below.

10 [0067] A general knowledge of the iSCSI and FC protocols is assumed. For more information on iSCSI refer to "draft-ietf-ips-iSCSI-09.txt," an Internet Draft and work in progress by the Internet Engineering Task Force (IETF), November 19, 2001, incorporated by reference herein. For more information about Fibre Channel (FC) refer to "Information Systems - dpANS Fibre Channel Protocol for SCSI," Rev. 012, December 4, 1995 (draft proposed American National Standard), incorporated by reference herein. In addition, both are further described in Provisional Application No. 60/325,704.

15 Storage Pools

[0068] As shown in Fig. 2, in its physical configuration, a system in accordance with an embodiment of the invention includes a switch 204 coupled to one or more servers 202 and to one or more physical devices 206, i.e., storage devices or subsystems. Each physical target is comprised of one or more logical units (LUs) 207. It is from these LUs that virtual targets will ultimately be formed.

20 [0069] However, before a virtual target can be created, or "provisioned," the switch needs to be "aware" of the physical storage devices attached and/or available for access by it as well as the characteristics of those physical storage devices. Accordingly, in one embodiment of the invention, when a storage device or an initiator device is connected to or registered with the switch, the switch must learn about the performance characteristics of the new device. In one embodiment, the switch includes a utility program, which can measure storage access time, data transfer rate, cache support, number of alternate paths to the device, RAID support, and allowable maximum commands for the LUs of

25  
30

the physical device. In some embodiments, once a device is connected to the switch, the utility program will automatically discover the device and automatically gather the required information without any user or other intervention. In some such embodiments, the switch will “discover” the addition/removal of a device when there is a disturbance or reset on the signal lines to the port. Once the device is “discovered,” various inquiries are sent to the device to gather information regarding performance characteristics. For instance, read/write commands can be sent to measure transfer rate or to check access time. Alternatively, in some embodiments, the obtaining of performance characteristics can be done by having an administrator enter the performance characteristics at a management station 210, wherein the characteristics can then be provided to a switch 204.

[0070] Based on the information gathered about the device, all of which is generally invisible to the end user, in one embodiment of the invention the switch classifies the device based on a policy. For example, devices with the best characteristics may be classified as Platinum devices. Those with intermediate performance characteristics as Gold or Silver devices. Those with the worst performance characteristics as Bronze devices. Of course, the types of policies that are defined are infinite and will vary amongst embodiments of the invention. Moreover, in some embodiments an administrator could further subdivide the policies, e.g., Platinum Building 1, Platinum Building 2, and assign resources to such subdivided policies. Nonetheless, an example of policies used in one embodiment of the invention are shown in Table 1 below:

**Table 1**

Policy Name	Platinum	Gold	Silver	Bronze
<b>PERFORMANCE PARAMETERS</b>				
Access time in milliseconds	>7	>10	>12	>15
Transfer rate in Megabytes/Sec	>30	>20	>15	>10
Max cache size in Megabytes	>32	>16	>8	>1
I/O per second rating	>3000	>2000	>1000	>500
Mbytes/second for backup	>8	>5	>3	>1
Mean Time Between Failure (MTBF) in years	>15	>10	>8	>5
RAID Level 0, 1, 2, etc. 0xEE = none	1	5	None	None
Maximum allowable commands	>100	>50	>25	—

[0071] As shown in Fig. 8, once a policy has been determined for a storage device, the LUs for the device are assigned to a storage pool 802, sometimes referred to herein as a “domain.” Since each storage device is comprised of one or more LUs, all the LUs of a particular storage device are assigned to the same pool. However, in one embodiment, each LU is considered by the switch as a separate storage node and each LU is described by an LU object in the SCC database 512. Thus, each pool has as members the LUs. In one embodiment, assignment to a pool is done independent of the protocol under which the physical storage device operates, e.g., iSCSI or Fiber Channel. As will be understood by those of skill in the art, each pool is defined in a switch by a listing for the pool of the LUs assigned to it, which listing is stored in the SCC database 512 in one embodiment. Such a listing may be comprised of pointers to the LU objects.

[0072] Generally each pool will be accessible only to users with particular characteristics. For example, a storage pool may be established for those users located in a Building 1, where the pool is entitled “Building 1 Shared Gold Storage Pool.” Another exemplary pool may be entitled

“Engineering Exclusive Silver Storage Pool” and may be exclusively accessible by the engineering team at a particular company. Of course an infinite variation of pools could be established and those described and illustrated are exemplary only.

5     **[0073]**         In addition, in an embodiment, there are two special pools: a  
“Default Pool” and a “No Pool.” A Default Pool allows access to anyone with  
access to the storage network. A “No Pool,” in contrast, is not generally  
accessible to users and is only accessible to the switch itself or to the system  
administrator. Once assigned to a pool, the LUs can be reassigned to different  
10     pools by the switch itself or by a system administrator. For instance, an LU may  
initially be placed in the No Pool, tested, and then later moved to the default  
pool or other pool.

#### Quality of Service and Service Level Agreements

15     **[0074]**         Service Level Agreements (SLAs) are sometimes used in  
network communications, but have not generally been used in the context of a  
storage network and have not been used in storage networks with Quality of  
Service (QoS) policies. By providing SLA/QoS, a user can select the conditions  
of storing and retrieving data. In one embodiment a QoS policy is defined by  
20     three elements: provisioning a virtual target, provisioning an initiator  
connection, and defining a user domain. Each is discussed below. Nonetheless,  
some embodiments may not require all three elements to define a QoS policy.  
For instance, some embodiments may only require provisioning a virtual target  
and provisioning an initiator connection, but not the user domain. Other  
25     embodiments may use different elements altogether to define a QoS policy.

#### Provisioning a virtual target

**[0075]**         Once the LUs for physical devices are in an accessible pool (i.e.,  
not the “No Pool”), then a virtual target can be created from those LUs. Once  
30     created, as shown in Fig. 9, the servers (and their respective users) will “see”

one or more virtual targets 902, each comprised of one or more extents 907, but they will not necessarily “see” the physical devices 206. An extent is a contiguous part of or a whole LU from a physical device. As shown in the example of Fig. 9, each extent in the example virtual target 902 is formed from entire LUs from several physical devices. “Extent” may still be referenced by an LUN from an initiator, such as a server, which doesn’t realize a target is “virtual.” The composition of the virtual targets, including protocols used by the LU is irrelevant to the server. However, as shown in Fig. 9, each virtual target is comprised of extents that map to the LUs of physical devices 206.

**[0076]** To provision a virtual target, a user will select several characteristics for the virtual target in one embodiment of the invention including:

- the size (e.g., in Gigabytes);
- a storage pool, although in one embodiment the user may select only from the storage pools which the user is permitted to access;
- desired availability, e.g., always available (data is critical and must not ever go down), usually available, etc.;
- the WWUI of the virtual target;
- a backup pool;
- user authentication data;
- number of mirrored members;
- locations of mirrored numbers (e.g., local or remote).

Still in other embodiments of the invention, different, additional, or fewer characteristics can also be selected.

**[0077]** The switch then analyzes the available resources from the selected pool to determine if the virtual target can be formed, and in particular the switch determines if a number of LUs (or parts of LUs) to meet the size requirement for the virtual target are available. If so, the virtual target is created with one or more extents and a virtual target object is formed in the SCC database identifying the virtual target, its extents, and its characteristics.

Examples of user-selected characteristics for four virtual targets are shown in Table 2 below:

**Table 2 - Virtual Target**

Virtual Target	A	B	C	D
size	1 TB	500 GB	100 GB	2 TB
storage pool	platinum	gold	bronze	bronze
availability	always	always	high	high
WWUI	drive A	drive B	drive C	drive D
backup pool	tape 1	tape 2	tape 3	tape 4
authentication data	connection ID and password	connection ID and password	password	password
# of mirrored members	3	2	2	1
locations of replicated sites	local	local	remote	none
Switching priority (One of 4) (if all else is equal, which target has priority)	1	2	3	4
Read Load Balance—on or off—when mirroring chosen	On	Off	Off	Off
Type of Media for backup (backup pool)	Fastest	Fast	Medium	Slowest
Mirroring—on or off	On	On	Off	Off
How many paths to storage from server (used for load balancing)	2	2	1	1
Path to storage via how many switches	2	2	1	1
Auto Migration to another target on excessive errors—on or off	Off	Off	On	Off
Physical storage—exclusive or shared	Exclusive	Exclusive	Exclusive	Shared
Virtual target—exclusive or shared	Exclusive	Exclusive	Shared	Shared
VPN on WAN connections	Yes	Yes	No	No

IP Precedence (DiffServ, RFC 2474)	Yes	Yes	No	No
MTBF	15 yrs.	10 yrs.	5 yrs.	5 yrs.

5     **[0078]**         In addition to provisioning a new virtual target, a switch in accordance with an embodiment of the invention can also modify existing virtual targets with new or different information or delete virtual targets when they are no longer needed.

10                     Provisioning an initiator connection.

**[0079]**         When a server or other initiator is connected to a switch and the initiator supports iSNS or SLP, in one embodiment the initiator will register itself with the switch, resulting in an initiator object stored in the SCC database. In other embodiments, however, the switch will include an access provisioning function which creates, updates, or deletes an initiator connection.

15           **[0080]**         In creating the access connection — the connection between the switch and an initiator (such as a server) — a user will specify various parameters shown for one embodiment in Table 3:

20     **Table 3 - Initiator Connection**

the server WWUI
connection detail, such as protocol (e.g., GigE or Fiber Channel)
exclusive or shared
25     source and destination IP addresses
minimum and maximum percentage of bandwidth
# of connections required by the server
access security
read only or read/write
30     VPN enabled

[0081] Some or all of the above information is saved in an initiator object stored in the SCC database. When the connection is removed, the initiator object will be deleted.

[0082] The switch, the management station, or other network management then creates a storage pool for the particular connection, specifying the LUs available to the initiator to form virtual targets.

#### User Domains

[0083] Like physical devices, virtual targets can be assigned to a pool accessible only to those with specified characteristics. Thus, like physical devices, virtual targets can be assigned to a user-specific domain (sometimes referred to herein as the User's Domain), a default domain (accessible to anyone), or a No Domain. Each domain will be identified, in one embodiment, by an object in the SCC database that includes a listing of all the virtual targets assigned to the domain. For virtual targets, the No Domain may include spare virtual targets, members of mirrored virtual targets, or remote virtual targets from another switch. Essentially, the virtual target No Domain is a parking place for certain types of virtual targets. For ease of description, when referring to virtual targets, pools will be referred to herein as "domains," but when referencing physical devices, pools will continue to be referred to as "pools." It is to be understood, however, that conceptually "pools" and "domains" are essentially the same thing.

[0084] Once an initiator connection is provisioned, as described above, a virtual target is provisioned that meets the initiator's requirements and placed into an accessible pool for the initiator or a previously provisioned virtual target is made accessible to the initiator, e.g., by moving the virtual target to the initiator's user domain from another domain such as the No Domain or Default Domain. (Note that either the virtual target or the initiator connection can be provisioned first — there is no requirement that they be provisioned in a particular order). Then, once an initiator requests access to the virtual target,

e.g., by sending a read or write request, both the virtual target object and initiator object are read from the SCC database and information regarding the initiator connection and virtual target is passed to the relevant linecard(s) for use in processing the requests.

5     **[0085]**         Examples of provisioning virtual targets are given with reference to Figs. 10a-d. Referring to Fig. 10a, assume there are physical devices having a total of 6 LUs — LU1, LU2, LU3, LU4, LU5, LU6 — coupled to a switch and all are placed in a pool accessible to two initiators X and Y the “X-Y User Pool.” If initiator X requires two virtual targets, then in one situation the LUs  
10     are provisioned to form virtual targets VT1 and VT2, where VT1 includes as extents LUs 1-3 and VT2 includes as extents LUs 4-6, where both VT1 and VT2 are placed in the server X user domain, thus allowing server X to access both virtual targets as shown in Fig. 10b. Server Y will not have access to either VT1 or VT2 since no virtual targets have been placed in the Y user  
15     domain. Alternatively, referring to Fig. 10c, if both server X and server Y require one virtual target, then VT1 and VT2 may be provisioned as before, but VT1 is placed in server X’s user domain while VT2 is placed in server Y’s user domain.

20     **[0086]**         If instead Y requires a mirrored virtual target M, VT1 and VT2 will be created as members of the virtual target M. VT1 and VT2 will be placed in the switch’s No Domain while M is made accessible to Y, as shown in Fig. 10d. As members of M, VT1 and VT2 are not independently accessible.

25     **[0087]**         In some embodiments of the invention, not only are devices and virtual targets coupled to one switch accessible to initiators, but virtual targets provisioned on another switch are accessible as well. Referring to Fig. 11, server X is coupled to switch A and server Y is coupled to switch B. VT1 is provisioned as part of server X’s domain in switch A while VT2 is provisioned as part of server Y’s domain in switch B. In addition, switch B is provisioned as an initiator to switch A, and switch A is provisioned as an initiator to switch  
30     B. In this manner, switch A can access VT2 via switch B, and switch B can

access VT1 via switch A. Accordingly, VT1, referred to here as VT1' since  
access is via switch B, can be included in server Y's domain, and VT2, referred  
to here as VT2', can be included in server X's domain (note that although the  
LUs of physical devices can belong only to one pool at a time, virtual targets  
can belong to more than one domain at a time). When X accesses VT2, switch  
B sees switch A as an initiator. Similarly, when Y is accessing VT1, switch A  
sees switch B as an initiator. In one embodiment, an administrator will make  
selected resources of switch B available to other switches, e.g., switch A, and  
vice versa. Alternatively, in some embodiments, certain domains may be  
defined to allow access to their resources by multiple switches.

#### Defining SLA

[0088] In one embodiment of the invention, access to a virtual target by  
an initiator will be provided in accordance with an SLA selected by a user of  
which the QoS policy is only a part. An example of some parameters that may  
be selected for an SLA by a user in one embodiment are shown in Table 6  
below:

**Table 4**

SLA Parameters	
ID of initiator (identifies initiator object)	
ID of virtual target (identifies virtual target object)	
ID of User Domain	
ID of extent getting provisioned	
Automatically increase size of virtual target — on or off	
Automatically increase size at what threshold	
Automatically increase what percentage of size	
Numbers of local mirrors (may be restricted to possible range — see Table 2)	
Local domain ID for each local mirrored member (may be restricted it to possible range — see Table 2)	

5	Numbers of remote mirrors (may be restricted to possible range — see Table 2)
	Remote domain ID (identified locally) for each remote mirrored member (may be restricted to Possible range — see Table 2)
	Define Error Threshold in event auto migration is On (see Table 2)
	Backup Enable (Disabled by default)
	Backup Schedule
	Pool ID for Backup LU

10 [0089] When a user agrees to an SLA, the user also selects a quality of service (QoS) policy. As described above, in one embodiment, the QoS policy is generally defined by virtual target (as provisioned), the initiator connection (as provisioned), and the User Domain. Accordingly, referring again to Table 4, above, the first three entries in the table — “ID of Initiator,” “ID of Virtual

15 Target” and “ID of User Domain” — will inherently describe the QoS policy since the attributes of the initiator connection and virtual target were defined when these items were provisioned. For example, the minimum and maximum bandwidth for the initiator connection has already been identified (see Tables 2 and 3). The User Domain assists in defining the policy by determining, for

20 example, if the initiator connection or virtual target connection is slower and forcing the QoS to the slower of the two. Of course, as mentioned above, the User Domain may not be necessary in all embodiments. As well, other embodiments may define an SLA using more, fewer, or different parameters than those shown in Table 4 above.

25

#### Figure 12

[0090] Fig. 12 summarizes the steps to provision the virtual targets and connections in order to be able to provide QoS in one embodiment. As shown, a switch in accordance with an embodiment of the invention discovers and

30 determines the characteristics of physical devices in communication with the switch, step 1202. The switch then classifies those devices, step 1204, and

associates those devices with a particular storage pool, step 1204. The switch will receive information for an initiator connection, step 1208, and will then provision the connection, step 1210, creating an object in the SCC database. The switch will also receive parameters for a virtual target, step 1212, and will provision the virtual target in accordance with those parameters, step 1214, if the resources are available, creating an object in the SCC database. Note that steps 1208-1214 can be performed in any order, the order shown in Fig. 12 being exemplary only. After the virtual target is provisioned, a user domain is created and the virtual target placed in the user domain or the virtual target is placed in a pre-existing user domain, step 1216. A user could also attempt to access a previously provisioned virtual target (hence, step 1214 may not be necessary for every connection). Finally, a switch in accordance with an embodiment of the invention receives SLA/QoS parameters, step 1218.

#### Objects

[0091] As discussed above, each virtual target, each initiator connection, and each physical device is identified in the SCC database with information included in an object for the respective entity. Each virtual target object and physical target object will include a listing of extents or LUs that comprise it. An example of a Virtual Target object, in one embodiment of the invention, includes the following information:

- entity type
- entity identifier
- managing IP address
- time stamp and flags
- ports
- domain information
- SCN bit map
- capacity and inquiry information
- number of extents

- list of extents
- extent locator
- virtual mode pages
- quality of service policy (e.g., the first three entries of Table 4)
- 5     • statistics - usage, error, and performance data
- SLA identifier

A physical target (or LU) object may include similar information.

10     **[0092]**         In the object, “entity type” will identify whether the entity is a virtual target or physical target. “Entity identifier” is, in one embodiment, a WWUI, which may be created by the user in some embodiments. The “managing IP address” indicates the address of the device through which the entity is configured, e.g., a management station. For instance, a virtual target is configured through a management station, which is accessed through the SCC in one embodiment of the invention.

15     **[0093]**         “Time stamp and flags” are used to track events such as when the virtual target or other entity was created or changed. Flags may be used to indicate various services or events in progress, such as copying of the data in a virtual target. “Ports” include a list of the ports through which the LU can be accessed and include information regarding the port names and linecard  
20     number, TCP/IP address or Fiber Channel 24-bit address, and whether the port is a primary or secondary port for the entity.

25     **[0094]**         “Domain information” includes the storage domain or pool to which the virtual target or entity belongs. “SCN bit map” indicates system change notification for the virtual target. “Capacity and inquiry information” indicates how big the virtual or physical target is as well as the inquiry information usually provided by a device vendor. For instance, inquiry information for a physical device will often identify its manufacturer whereas inquiry information for a virtual target will often identify the switch that created the virtual target.

[0095] Each LU of a physical device is comprised of one or more contiguous pieces of storage space called an extent, which are used to form the virtual targets. Accordingly, "number of extents" identifies how many extents form the virtual target. "List of extents" identifies each of the extents, in one embodiment, by an offset and a size. For example, a 10GB virtual target comprised of three extents may identify the extents in the "list of extents" as shown in Table 5:

**Table 5**

extent	offset (virtual target)	size
1	0	2GB
2	2GB	5GB
3	7GB	3GB

10

[0096] "Extent locator" identifies exactly where the extents are located, i.e., on which physical devices. For example, the above 10GB, 3-extent virtual target may have the following extent locator:

15

**Table 6**

extent	storage device	offset (physical device)
1	2	5GB
2	1	3GB
3	3	15GB

20

[0097] In this example using both Table 5 and Table 6, it can be determined that the first extent of the virtual target is mapped to physical storage device 2 (Table 6) starting at an offset of 5GB (Table 5) and extending for 2GB (Table 5). The second extent (Table 5) is mapped to physical storage device 1 (Table 6) starting at an offset 3GB (Table 6) and extending for 5GB (Table 5). And finally, the third extent is mapped to physical storage device 3 (Table 5) starting at an offset 15GB (Table 6) and extending for 3GB (Table 5).

25

[0098] If the virtual target is mirrored, as it may be in some embodiments, every member of the mirrored virtual target will have an identical extent list, although the extent locators will be different.

5 [0099] "Virtual mode pages" identify the mode pages frequently found in SCSI commands as will be understood in the art. This information includes the block transfer size, immediate data support, or any unique information that application software with SCSI-mode-page commands can set and retrieve.

10 [0100] "Quality of service policy" determines the service attributes for the virtual target and is selected at the time of provisioning of the virtual target. In one embodiment, Quality of Service policy will be defined using the identifiers found in the first three entries of Table 4.

15 [0101] "Statistics" are collected at run time of the virtual target by the switch in one embodiment of the invention. They may include usage, error, and performance data in one embodiment of the invention, and are further discussed below.

[0102] The "SLA identifier" identifies an SLA object for information regarding the SLA.

#### Statistics

20 [0103] A switch in accordance with an embodiment of the invention also collects statistics. In one embodiment, for each connection from one initiator to one virtual target, the following information is collected by the SPU of the linecard connecting to the initiator:

- 25 1. Total read access (number of read requests);
2. Accumulated read transfer bytes (total number of bytes read from storage);
3. Accumulated read response time (time from receiving request to getting a response);
4. Total write access (number of write requests);
- 30 5. Accumulated write transfer bytes;

6. Accumulated write response time;
7. Accumulated recoverable errors;
8. Accumulated unrecoverable errors.

5     **[0104]**       The CPU on each linecard periodically requests the statistics from the SPU. The SPU responds by returning the data. The SPU then resets the data to zero and resumes collection.

**[0105]**       Based on the collected data, the CPU maintains the following statistics:

- 10           1. Average read access rate;
2. Maximum read access rate;
3. Average read transfer rate;
4. Maximum read transfer rate;
5. Minimum read response time;
6. Average read response time;
- 15           7. Maximum read response time;
8. Average write access rate;
9. Maximum write access rate;
10. Average write transfer rate;
11. Maximum write transfer rate;
- 20           12. Minimum write response time;
13. Average write response time;
14. Maximum write response time;
15. Recoverable errors per billion of requests;
16. Unrecoverable errors per billion of requests.

25     **[0106]**       After some pre-selected time period in one embodiment, the CPU forwards the statistics to the SCC and updates the relevant VTDs (stored in the SPUs). In another embodiment, the SCC will request the statistics from the CPU, and the CPU will provide them to the SCC. In some embodiments, the SCC will also reset its statistics periodically, e.g., weekly, to ensure that data  
30     is accurate and not over-accumulated.

Enforcing QoS

[0107] The minimum percentage of the initiator connection bandwidth is guaranteed by the QoS in one embodiment. Hence, in such an embodiment when multiple initiators are provisioned on a single port, the sum of all minimum bandwidths of all initiators must be less than or equal to 100%. In contrast, the maximum percentage provides the allowable use of the connection when there are no other contending users on the same connection. Thus, the sum of maximum percentages of bandwidths of all initiators can exceed 100% of the bandwidth of the connection. When they do, the defined switching priority (see Table 2) determines which initiator gets scheduled first.

[0108] In a conventional communications network (as opposed to a storage network), QoS is used to ensure that users get the percentage of data bandwidth of a connection that they paid for. It allows time-sensitive data such as audio and video to experience only acceptable interruptions by either negotiating a reserved data bandwidth before transmission or giving the time-sensitive transmission a higher priority in a congested situation. The QoS is enforced by prioritizing the switching traffic even at the expense of dropping packets.

[0109] However, dropping a request in a storage system is unacceptable, unlike conventional network communication system, where a request may include one or more packets. In one embodiment, a request includes all packets sent back and forth from initiator to target until the request is complete, e.g., an iSCSI command PDU, an iSCSI R2T, an iSCSI write data PDU, and an iSCSI response PDU will form a single request. For a storage switch in accordance with an embodiment of the invention, the data bandwidth, in one embodiment, is calculated by the number of requests per second multiplying by the average transfer size of the request. For example, if the average transfer size is 8KB, with 1000 requests per second, the bandwidth for the storage device will be 8MB/sec (or 80Mb/sec). But since a switch has no control of the average transfer size of the request, enforcing the QoS for storage access is to control the

number of concurrently allowed requests per second. Thus, if too many requests are sent from an initiator, the number of concurrent requests must be reduced. In one embodiment, in a worst case only one request can be sent by an initiator at a time.

5     **[0110]**         A virtual target supports a maximum number of concurrent requests. An initiator accessing multiple virtual targets can have a maximum number of requests sent that is equal to the sum of the maximum number of requests for all of the virtual targets it is accessing. But, when multiple  
10     available are shared among the initiators, being prorated according to the respective QoS parameters of minimum percentage of bandwidth. For instance, if two initiators share access to a virtual target that can accommodate 100 concurrent requests, and initiator 1 gets a minimum of 70% of the bandwidth while initiator 2 gets a minimum of 30% of the bandwidth, then *initially*  
15     initiator 1 can send 70 requests and initiator 2 can send 30 requests. Nonetheless, because each initiator will have its own request size, a large request size may consume greater bandwidth and crowd out other initiators of smaller transfer sizes. Thus, adjustment of allowable requests by each initiator in order to guarantee a bandwidth range is performed in one embodiment as  
20     follows.

**[0111]**         The traffic managers (TMs) 608 (Fig. 6) in both ingress and egress linecards monitor the transfer bandwidth of different connections. The TM also schedules delivery based on QoS parameters. Thus, the TM guarantees that each shared connection gets its minimum bandwidth and is limited by its  
25     maximum bandwidth — in other words, the TM assures that each connection is within a specified range. To do so, in one embodiment, as packets accumulate inside the TM buffer 612, such accumulation will indicate that an initiator has exceeded its limitations. The TM will send a control message to the SPU indicating that the offending initiator should slow its connection. After  
30     receiving such a message, the SPU will reduce the number of allowable requests

to the offending initiator while the number of allowable requests to the initiator that was receiving a smaller share would be increased. In one embodiment, notification of the number of requests available to a server may occur in the MaxCmdSN field of an iSCSI PDU

5     **[0112]**         For example, an initiator A and an initiator B both have as their minimum bandwidth 50% of a shared initiator connection. Using a transfer size of 100KB, initiator A sends 800 requests per second thus getting 80MB per second of bandwidth on the connection. Using a transfer size of 4K, initiator B sends 2000 requests per second, but gets only 8MB per second of bandwidth.

10    Thus, if the maximum bandwidth allowed for initiator A is 70MB per second, the switch must reduce the number of requests from initiator A to reduce its requests to 700 per second to obtain 70MB per second. Accordingly, the ingress traffic manager 608<sub>i</sub> will report to the ingress SPU that initiator A has exceeded its maximum and packets are accumulating in the buffer 612<sub>i</sub>. The SPU, in

15    receiving the message, will reduce the number of allowable requests to A and increase those to B. Thus, initiator B will be able to send more requests on the connection. It should be noted that when the initiator is not maximizing the use of its allowable requests to even reach its minimum percentage bandwidth, no adjustment will be necessary. Further, because initiator B is not currently

20    demanding 50% of the connection, initiator A is free to use up to (but not to exceed) its maximum allowed bandwidth.

**[0113]**         Similarly, if two initiators on two different connections are sharing a single virtual target, the prorated request numbers for each initiator are adjusted when the TM 608<sub>e</sub> on the egress linecard detects unfair bandwidth uses

25    between the two initiators. It will detect such unfair bandwidth usage when the offending initiator has packets accumulated in the buffer 612<sub>e</sub>.

**[0114]**         When the connection is not shared and becomes congested due to the physical storage device itself being busy, the egress TM 608<sub>e</sub> will inform the PPU because packets are accumulating in the buffer 612<sub>e</sub>. Again, the SPU will

30    then reduce the number of allowable requests to slow down the initiator(s).

[0115] The switch will also match the bandwidth between the initiator and the storage device. For example, to support an initiator having a minimum of 100% of a 1Gb connection, no other virtual target can be allocated on the storage connection. But when an initiator only requires 50% bandwidth of the connection, the remaining 50% can be allocated to another virtual target.

[0116] Finally, when everything else is equal, the priority of a connection determines which command gets delivered first by the switch traffic manager of a linecard.

[0117] Table 7 below summarizes the QoS enforcement discussed herein for one embodiment.

**Table 7**

initiator ingress port	target egress port	detection	actions
not shared	not shared	egress buffer threshold	reducing allowable requests
shared	not shared	ingress buffer threshold	reducing allowable requests from offending initiators
not shared	shared (shared target)	egress buffer threshold	redistribute allowable requests to different initiators
not shared	shared port (different targets)	egress buffer threshold	reducing allowable requests to offending initiator
shared	shared	ingress and egress buffer threshold	treat each virtual target separately as the above four cases

[0118] For the first situation, where an initiator ingress port is not shared and the target egress port is not shared, congestion will often be caused by busy physical target devices and will generally be detected when an egress buffer threshold is exceeded (the egress buffer will be backed up beyond an acceptable

point). Thus, appropriate action is to reduce the allowable number of requests from the initiator.

5 [0119] In the second situation, the shared initiator ingress port is shared by initiators that are accessing different targets on different ports, so that the target egress port is not shared. Excessive bandwidth use by one of the initiators is detected in the ingress buffer by determining if a threshold has been exceeded, causing the buffer to back up beyond an acceptable point. Appropriate action is to reduce the allowable number of requests from the offending initiator.

10 [0120] In the third situation, the initiator ingress port is not shared but the target egress port is shared, indicating that the same target is accessed by different initiators from different ports. Excessive bandwidth usage caused by an excessive number of requests by one of the initiators will be detected in the egress buffer. Appropriate action is to redistribute the number of allowable requests from the different initiators, e.g., decrease the number of requests 15 allowed one initiator while increasing the number of requests to the other initiator.

20 [0121] In the fourth situation, the initiator ingress port is not shared but the target egress port is shared, but in this instance different targets are accessed on the same egress port by different initiators. In such a circumstance, excessive bandwidth is detected in the egress buffer where each target is given a percentage of the connecting bandwidth. Appropriate action to take in such circumstances is to reduce the number of allowable requests to the offending initiator.

25 [0122] Finally, the fifth situation indicates a shared initiator ingress port and a shared target egress port. In such a situation, there is a two-tiered decision: first to ensure that each virtual target is getting its allocated percentage of bandwidth, and then second, to prorate the allowable number of requests to different initiators. Such decision making takes place in both ingress and egress 30 buffers by looking to see if the buffer thresholds have been exceeded.

Appropriate action is to treat each virtual target separately as is done in the above four circumstances and to reduce the number of requests as required.

[0123] As should be understood, Table 7 is illustrative only. In other embodiments, other actions could occur to enforce QoS and other situations could occur that are not described above.

#### Load balancing

[0124] Load balancing is utilized in one embodiment and occurs by selecting a path dynamically to reach a target device faster when more than one path is available to the target device. Load balancing is done dynamically (as opposed to statically, at fixed time intervals) on every port in the switch and for each request by utilizing the SPU processing power on each port.

[0125] Failover is a special case of load balancing and utilized in some embodiments of the invention. Failover will occur when one member of a mirrored target becomes unavailable or one path becomes unusable to a target that is accessible by multiple paths – in either case, the other member is accessed or the other path is utilized.

[0126] In a switch in accordance with an embodiment of the invention, the switch performs two different types of actions related to load balancing:

1. Referring to Fig. 13b, if the virtual target is mirrored, the switch will steer initiator read requests to one of the mirrored members by selecting the member of the mirrored virtual target with the shortest average response time; and
2. Referring to Fig. 13a, if there is more than one path to an LU, the switch will steer requests to the LU on the path with the shortest average response time. However, in one embodiment, this load balance action is only performed when the multiple paths are connected from the target LU to the same SPU, although other embodiments may not have such a requirement.

[0127] In some embodiments, a switch will also support a “pass-thru” configuration. In such an embodiment, the virtual target is the physical target itself, and all commands “pass-thru” the switch without interpretation – e.g., without virtualization or translation. In such embodiments, all load balance  
5 functions are handled by the server itself.

[0128] More specifically, for load balancing, using the statistics collected as discussed above, a switch in accordance with the invention tracks the average response time of each target, including the response time of each of the members of a mirrored virtual target. The relevant statistics are stored in  
10 each VTD, which is periodically updated by the CPU. On a read operation, the SPU (referring to the VTD) then selects the path with the shortest average response time and forwards the request on that path or it selects the mirrored member with the shortest average response time and forwards the request to that member. Note that with mirrored targets, a selection amongst mirrored  
15 members would not be performed for write operations since writes will be made to all members of a mirrored virtual target. When there is no clear advantage of one path over the other, or one mirrored member over the other, the commands are sent to the various paths/members alternately.

[0129] In one embodiment of the invention, multiple concurrent  
20 connections will only be used for iSCSI devices, as Fibre Channel does not currently support such multiple concurrent connections. However, other embodiments using other protocols may also support multiple concurrent connections.

[0130] It should be understood that the particular embodiments  
25 described above are only illustrative of the principles of the present invention, and various modifications could be made by those skilled in the art without departing from the scope and spirit of the invention. Thus, the scope of the present invention is limited only by the claims that follow.